Vietnam J Comput Sci (2015) 2:257–265
DOI 10.1007/s40595-015-0050-0

REGULAR PAPER

(1,7,4) Data hiding scheme for searching in hidden text with automata

Le Quang Hoa[1] · **Nguyen Huy Truong**[1] · **Cheonshik Kim**[2] · **Ching-Nung Yang**[3]

Received: 29 December 2014 / Accepted: 1 August 2015 / Published online: 9 August 2015

Abstract In this paper, we present a new data hiding scheme (1,7,4) for grayscale images, where 7 is the number of pixels in each block of the image, 4 is the number of secret bits which can be hidden in each block with restriction that at most one pixel is changed, and the gray value is changed at most 3 from the original one. As shown in the paper, this reaches the theoretical limit MSDR1 of hidden bits in each 7-block of images. A hiding scheme (2,14,8) is obtained as a direct consequence, and one application of this scheme is given to solve the problem of searching in hidden texts in stego-images. To solve this problem, finite automata technique with fuzzy approach is introduced.

Keywords Grayscale image · Steganography · Hiding scheme · MSDR · Fuzzy approach

✉ Cheonshik Kim
mipsan@paran.com

Le Quang Hoa
hoalqbk@gmail.com

Nguyen Huy Truong
truong.nguyenhuy@hust.edu.vn

Ching-Nung Yang
cnyang@mail.ndhu.edu.tw

[1] School of Applied Mathematics and Informatics, Hanoi University of Science and Technology, 1 Dai Co Viet Road, Hanoi, Vietnam

[2] Department of Digital Media Engineering, Anyang University, Anyang, Republic of Korea

[3] Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien, Taiwan, ROC

1 Introduction

Data hiding [1–19] is interested in several authors because data hiding can be used for copyright, secret communication, military fields, and hospital, etc. Data hiding is referred to as a process to hide data (representing some information) into cover media. It is possible to conceal hiding data in an image because the human visual system has low sensitivity to small changes and the digital media is high flexibility, and one can easily change LSB of an image with low perceptibility [1]. For this reason, many researchers created various data hiding schemes for a lot of interesting applications in various fields. Conceptually, data hiding is similar to watermarking, but watermarking is based on in frequency domain and it is very complex and takes more time and makes more changes in the cover image. The merit of watermarking is a strong various digital attack such as changing format and cutting an image [2].

On the other hand, data hiding is based mainly on spatial domain instead of the frequency domain. Therefore, this scheme does not appropriate to hide copyright in an image since the spatial domain is originally very fragile. Machado proposed EzStego [3] schemes for palette-based images in 1997. In this method, palette is first sorted by luminance to minimize the perceptual distance between consecutive colors. For embedding data, this method is based on the distance of luminances.

Palette modification scheme proposed by Fridrich [4] can iteratively embed one message bit into each pixel in a palette-based image. S-Tools [5] does not drastically change the color value of an image as the number of colors are reduced from 256 to 32 while maintaining the image quality. S-Tools manipulates the palette to produce colors that have a difference of one bit. Bit Plane Complexity Segmentation (BCPS) [6] was based on the simple idea that

the higher bit planes could also be used for embedding information.

In 2006, Mielikainen proposed LSB matching revisited technique [7], which is hiding two secret bits in a pair of cover pixels, only one cover pixel needs to be modified. EMD scheme [8] is possible to embed secret digit by modifying one pixel in a $(2n + 1)$—ary notational system [9]. Chang et al. [10] and Zhang et al. [11] proposed (7,4) Hamming code for data hiding, which improves on the Hamming+1 scheme.

The majority of steganographic utilities for the camouflage of confidential communication suffers from fundamental weaknesses. On the way to more secure steganographic algorithms, the development of attacks is essential to assess security. In [5], the visual attacks presented here exemplify that at least EzStego v2.0b3, Jsteg v4, Steganos v1.5, and S-Tools v4.0 suffer from the misassumption that least significant bits of image data have uncorrelated noise.

We present a new good (1,7,4) hiding scheme in which $N = 7$ is the number of pixels in each block and 4 is the number of hidden bits which can be embedded in each block by changing at most one pixel, the changed gray value is not larger than 3. Therefore, our proposed scheme is safe from the attacks, because the ratio of changed pixel for data hiding is less than 1/7. This scheme (1,7,4) is directly extended to a (2,14,8) hiding scheme which in its turn shows advantages to resolve the problem matching in hidden secret text in stego-images without a secure environment for end-users. For this purpose, we are using a modification of text matching technique by automata theory.

The relationship between module and covering code technique in data hiding area is considered in [12–14]. The hiding scheme (2,14,8) provides us a suited way to build an algorithm for searching hidden text by automata technique which now exists in several sources [15–18]. In this scheme, we introduce a fuzzy approach to searching hidden text by automata and introduce a hidden text searching technique in stego-images without revealing extracted text using our hiding scheme (2,14,8). The rest of this paper is organized as follows. In Sect. 2, we review MSDR and hiding schemes using Z_q -modules. In Sect. 3, we propose (1,7,4) scheme for 8-bit grayscale images. In Sect. 4, we propose how to search in hidden text with fuzzy automata approach. In Sect. 5, we explain the experimental results. Section 6 presents our conclusions.

2 MSDR and hiding schemes using Z_q-modules with small q

2.1 Theoretical limit MSDR for hiding schemes

In this section, given an image G, we only concentrate on a fixed block $F = (F_1, F_2, \ldots, F_N)$ of N pixels of G, and F is considered as a vector of dimension N). In F, each entry F_i can be understood as a pixel whenever the index i is referred. In binary images, $q = 2$. In the general case $q \geq 2$ for color images or grayscale images.

When secret bits can be embedded in each block F by changing at most k entries where $k = [1,2]$, it call a k-hiding scheme. Each new block F after changing pixels of F is called a configuration. The k-Maximal Secret Data Ratio (in theoretical limit) presents the upper bound number of embedded bits in each block F of N pixels by changing colors of at most k pixels in F.

In the case $k = 1$, we change at most one element in F, with N elements, $1 + (q - 1)N$ ways can be taken. This means that for any 1-hiding scheme, we can hide at most $MSDR_1 = \lfloor \log_2(1 + (q - 1) \cdot C(N, 1)) \rfloor$ secret bits in each block F.

In the cases $k = 2$, we have $MSDR_2 = \lfloor \log_2(1 + (q - 1) \cdot C(N, 1) + (q - 1)^2 \cdot C(N, 2)) \rfloor$. For example, with $N = 7, q = 4, MSDR_1 = \lfloor \log_2(1+3 \cdot 7) \rfloor = 4$. That means most 4 secret bits can be hidden in F by theoretical limit. This is reached by our scheme (1,7,4) which is introduced below.

2.2 Module method

First, let us recall some properties and notions. Each (right) module M over the ring Z_q is an additive abelian group M with an element $m \cdot t$ in M so that some properties are satisfied. Let $Zq = \{0, 1, \ldots, q - 1\}$. This scheme requires following basic properties.

(P1) $m \cdot 0 = 0. m \cdot 1 = m.$
(P2) $m + n = n + m$ for all m, n in M.
(P3) $m \cdot (t + l) = m.t + m.l$ for all m in M, t, l in Z_q.

Definition 1 Given a natural number $v > 0$, a subset $U \subseteq M - \{0\}$, we call U a v-base of M if for any $x \in M - \{0\}$, x can be presented by a linear combination of at most v elements in U. That means there exist n elements $u_1, u_2, \ldots, u_n$ in U, $n < v$, together with $t_1, t_2, \ldots, t_n$ in Z_q such that $x = u_1 \cdot t_1 + u_2 \cdot t_2 + \cdots + u_n \cdot t_n$.

It is obvious that $U = M - \{0\}$ is the unique 1-base. In this part, the main interest is the case $v = 1$ for binary images (according to the characteristic $q = 2$ and $M = Z_2^n = Z_2 \times Z_2 \times \cdots \times Z_2$ is the n-fold cartesian product of Z_2, which can be seen as a (right) Z_2-module. The addition in Z_2 can be seen as the operation XOR (exclusive—OR) on bits. Each element $x = (x_1, x_2, \ldots, x_n)$ in M can be presented as an n-bit string $x = x_1 x_2 \ldots x_n$, with operations defined as follows:

(D1) $x + y = z_1 z_2 \ldots z_n$ where $z_i = x_i \oplus y_i, i = 1, \ldots, n,$
For $x = x_1 x_2 \ldots x_n, y = y_1 \ldots y_n$ in M, k in Z_2,
(D2) $x \cdot k = z_1 z_2 \ldots z_n$ where $z_i = x_i \cdot k = x_i$ AND k.

Given a binary image G, we set $C_G = Z_2 = \{0, 1\}$ as the set of two colors of G. The color changing function Next: $Z_2 \to Z_2$ is given by: $c' = c + 1 = Next(c)$, for all c in Z_2 and changing a color c means that c is replaced by $c' = Next(c)$.

2.2.1 Application of 1-bases for 1-hiding schemes in binary images

Let $U = M - \{0\}$ be the 1-base of the Z_2 module $M = Z_2^n$, $|U| = 2^n - 1$. Consider any binary block $F = (F_1, F_2, \ldots, F_s)$ of a given binary image G, and binary secret key $K = (K_1, K_2, \ldots, K_s)$, $F_i, K_i \in Z_2$, $i = 1, \ldots, N$. Suppose $s \geq 2^n - 1$. Then, we can assign a surjective function $h : \{1, 2, \ldots, N\} \to U$ as a weight function of elements in F where $h(i) \in U$ is considered as the weight of F_i. We show that for any secret element $d \in M$ where d can be embedded in F by changing colors of at most 1 element in F as the following 1-hiding scheme.

A. Hiding a secret element d

Step (0) Given a secret key as a binary vector $K = (k_0, k_1, \ldots, k_N)$, $k_i \in Z_2$.
Compute $T = (t_0, \ldots, t_N)$ by taking $T_i = F_i + k_i$ (in Z_2) for all i.
Denote $T = F \oplus K$;

Step (1) Calculate $m = S[F, T]$ where $S[F, T]$ is the sum $\oplus_{1 \leq i \leq N} h(i) \cdot T_i$, by taking operations on the Z_2 module M.

Step (2) Compare m and d:

- Case $m = d$: keep F intact;
- Case $d \neq m$: then find $d - m = a$, for some $a \in M - \{0\}$.

Since h is surjective, there exists F_q in F, such that $h(q) = a = d - m$. Then, change the color F_q to new color $F_q = Next(F_q) = F_q + 1$.

B. Extracting the secret element embedded in F

Step (1) Calculate $u = S[F, K]$;

Step (2) Return u as the secret element d embedded in S (that is $u = d$).

Correctness of the method is shown by.

Theorem 1 [13] *The element u extracted in step 1 of the extracting stage A is exactly the secret element d hidden into S in the hiding stage B above.*

Example 1 We can build a hiding scheme (1,7,3) for each 7-pixel block $F = \{p1, \ldots, p7\}$ of any binary image G as following:

- Set $U = 001, 010, 100, 100, \ldots, 111$ is a 1-base containing 7 elements of the module $M = Z_2^3$, in U these elements can be considered as binary presentation of the natural numbers $1, 2, \ldots, 7$, simply we write $U = \{1, 2, \ldots, 7\}$.
- We can assign U as the weight set of each 7-pixel block $F = \{p_1, \ldots, p_7\}$, each j in U is the weight of the pixel p_j. Therefore, we can use it to hide any 3 secret bits which is considered as an element of Z_2^3 by changing at most 1 pixel.

Remark 1 The F5 hiding scheme introduced in [20] is nothing but an 1-hiding scheme using the Z_2-module $V_n = Z_2^n = Z_2 \times \cdots \times Z_2$.

3 (1,7,4) schemes for 8-bit grayscale images

3.1 Hiding scheme (1,7,4)

For a given 8-bit grayscale image G, at first, we split its pixels into several disjoint 7-block of pixels. In each block, say $F = (p_1, p_2, \ldots, p_7)$, at first we set $V_F = (v_1, \ldots, v_7)$ the binary vector of LSB bits $v_1, \ldots, v_7$ of these elements, that is $v_i = p_i \ mod \ 2$ which is considered as an element of Z_2, for $i = 1, \ldots, 7$.

Using V_F as a 7-block of 7 binary pixels, we can apply the method mentioned in Example 1 to embed any 3-bit string $d = d_3d_2d_1$ in V_F by changing at most one place, say p_k in V_F. For such a case, we replace the pixel p_k in F by some ways, $p_k + 1$ or $p_k - 1$or $p_k + 3$ or $p_k - 3$ so that in the new vector V_F the new LSB v_k of p_k is flipped. In the case V_F is not changed, we can remain F or change any pixel p_k by the new $p_k + 2$ or $p_k - 2$ so that V_F is not changed. By this ways, any given 3-bit string d can be embedded in F(new) since from V_F we can extract d by the method given above. The complicated situation of choosing which changes, p_k by $p_k + 1$ or $p_k - 1$, $p_k + 3$, $p_k - 3$, $p_k + 2$, $p_k - 2$, is taken carefully so that we can embed one more bit d_4 in F, for any free bit d_4. Concretely, we extend the scheme (1,7,3) to the hiding scheme (1,7,4) to hide 4 bits in F by changing at most 1 pixel of F as follows.

3.1.1 Hiding phase c into F

Given a secret key assigned to F as a binary vector $K = (k_1, \ldots, k_7)$, $k_i \in Z_2$.

Step (1) Calculate $V_F = (v_1, \ldots, v_7)$ where $v_i = p_i \ mod \ 2$, for $i = 1, \ldots, 7$.

Step (2) Calculate $T = V_F \oplus K$ by taking $t_i = v_i \oplus k_i$ (addition in Z_2).
Step (3) Compute $m = S[V_F, T] = \oplus_{1\le i\le N} i \cdot t_i$ (a sum in Z_2-module $M = V_3$);
Step (4) Calculate the bit $x = (\Sigma_{1\le i\le 7}\lfloor p_i/2\rfloor)\ mod\ 2$;
Step (5) Compare m and d:

(5.1) Case $m = d$ and $x = d_4$: keep F intact;
(5.2) Case $m = d$ and $x \neq d_4$: choose any one pixel p_k in F and replace p_k to p'_k (we should choose $p_k \neq 0{,}255$ if it exists) so that $x' = (\Sigma_{1\le i\neq k\le 7}\lfloor p_i/2\rfloor + \lfloor p'_k/2\rfloor)\ mod\ 2 = x \oplus 1 = d_4$ by:
(5.2.1) if $p_k \le 1$ or p_k is even and $p_k < 254$: put $p'_k = p_k + 2$;
(5.2.2) if $p_k \ge 254$ or p_k is odd and $p_k > 1$: put $p'_k = p_k - 2$;
(5.3) Case $d \neq m$ and $x = d_4$: choose one pixel p_k in F so that $v_k = d \oplus m$ and replace p_k to p'_k so that $x' = (\Sigma_{1\le i\neq k\le 7}\lfloor p_i/2\rfloor + \lfloor p'_k/2\rfloor)\ mod\ 2 = x = d_4$ by:
(5.3.1) if p_k is even: put $p'_k = p_k + 1$;
(5.3.2) if p_k is odd: put $p'_k = p_k - 1$;
(5.4) Case $d \neq m$ and $x \neq d_4$: choose one pixel p_k in F so that $v_k = d \oplus m$ and replace p_k to p'_k so that $x' = (\Sigma_{1\le i\neq k\le 7}\lfloor p_i/2\rfloor + \lfloor p'_k/2\rfloor)\ mod\ 2 = x \oplus 1 = d_4$ by:
(5.4.1) if p_k is even: put $p'_k = p_k + 1$;
(5.4.2) if p_k is odd: put $p'_k = p_k - 1$;
(5.4.3) $p_k = 0$: put $p'_k = p_k + 3 = 3$;
(5.4.4) $p_k = 255$: put $p'_k = p_k - 3 = 252$;

Step (6) Return F(new) from F by changing p_k to p'_k if it is needed;

Then, F(new) contains c as 4 embedded bits which can be extracted in the following phase.

3.1.2 Extracting phase to get the secret element c embedded

Step (1) Calculate $V_F = \{v_1, \ldots, v_7\}$ where $v_i = p_i\ mod\ 2$, for $i = 1, \ldots, 7$.
Step (2) Calculate $T = V_F \oplus K$ by taking $t_i = v_i \oplus k_i$.
Step (3) Compute $m = S[V_F, T] = \oplus_{1\le i\le N} i \cdot t_i$ (sum in Z_2-module $M = V_3$) as a 3-bit string; Present $m = d_3d_2d_1$;
Step (4) Calculate $x = (\Sigma_{1\le i\le 7}\lfloor p_i/2\rfloor)\ mod\ 2$ and put $d_4 = x$, $c = m + 8x(= d_4d_3d_2d_1)$;
Step (5) Return(c);

The correctness of this scheme is given by

Theorem 1 *In the extracting phase 3.1.2, the return value is the exact 4-bit string c proceeded to hide in the new F block of 7 pixels after taking the hiding phase 3.1.1.*

Proof It is easily seen that:

After the steps 5.1 and 5.2, the values v_i in V_F are not changed, hence it remains $m = d$, but the value x after the step 4 always equals to the bit d_4. After the cases 5.3 and 5.4, the new values v_i in V_F are changed so that $m = d$ in extracting phase, and the value x after the step 4 always equals to the bit d_4, hence, $c = m + 8x = d_4d_3d_2d_1$ as claimed. □

Remark 2 (i) In the above scheme, in hiding phase, each pixel in F is changed with at most 3 other values and the differences of these values to original value do not exceed 3, hence applying $MSDR_1 = \lfloor \log_2(1 + (q - 1)\cdot C(N, 1))\rfloor$ with $q = 4$ and $N = 7$ provides us $MSDR_1 = 4$, it is exactly the bit number of c. Therefore, the performance of hiding scheme (1,7,4) attains limit $MSDR_1$ of data hiding ratio of 1-hiding scheme.

(ii) By the hiding scheme (1,7,4), we obtain directly the hiding scheme (2,14,8) by applying a couple of two consecutive 7-pixel blocks as a 14-pixel block and we can change at most two pixels, each pixel in each sub-block, by the way above to hide 8 bits, each 4 bits hide into one sub-block of 7 pixels. With this scheme, we have $MSDR_2 = \lfloor \log_2(1 + 3\cdot 14 + 32\cdot 13\cdot 14/2)\rfloor = \lfloor \log_2 862\rfloor = 9$ as the theoretical limit of the number of bits which can be hidden in each 14-block. Here, we gain 8 hidden bits, smaller than the limit one bit. A question arisen naturally: whether or not there exists a hiding scheme (2,14,9).

(iii) In application, to prevent from brute-force attacks, it is dangerous if we use only one binary key vector of 7 values. Instead of this, we can use a long binary of $7 \times 20 = 140$ bits and use this as 20 binary key vectors of 7 bits: $K_i = (k_{i1}, k_{i2}, \ldots, k_{i7})$ with $i = 1, 2, \ldots, 20$ and use consecutively these to assign with consecutive 7 blocks of pixels of a given image G.

Example 2 Let $F = (p_1, \ldots, p_7)$ with $p_1 = 12$, $p_2 = 8$, $p_3 = 0$, $p_4 = 25$, $p_5 = 84$, $p_6 = 255$, $p_7 = 45$, $K = (0, 1, 1, 0, 0, 1, 0)$ is the binary vector key assigned with F. In the hiding phase, $V_F = (v_1, \ldots, v_7)$ with $v_1 = 0$, $v_2 = 0$, $v_3 = 0$, $v_4 = 1$, $v_5 = 0$, $v_6 = 1$, $v_7 = 1$. Then, $T = V_F \oplus K = (0, 1, 1, 1, 0, 0, 1)$ and $m = S[V_F, T] = \oplus_{1\le i\le N} i \cdot t_i = 2 \oplus 3 \oplus 4 \oplus 7$ or in binary presentation $m = 010 \oplus 011 \oplus 100 \oplus 111 = 010$; $x = (\lfloor 12/2\rfloor + \lfloor 8/2\rfloor + \lfloor 0/2\rfloor + \lfloor 25/2\rfloor + \lfloor 84/2\rfloor + \lfloor 255/2\rfloor + \lfloor 45/2\rfloor)\ mod\ 2 = (6+4+0+12+42+127+22)\ mod\ 2 = 1$; Consider some typical cases of c - the 4-bit string needed to embed into F:

(a) $c = 1010$: in hiding phase, $d_4 = c\ div\ 8 = 1$; $d = c\ AND\ 111 = 010 = m$ and $x = d_4$. In this case, F

is kept intact. In the exacting phase, $c = m + x \cdot 8 = 010 + 1000 = 1010$ is recovered.

(b) $c = 1001$: then $d_4 = 1 = x$ and $d = 001 \neq m$, set $u = d \oplus m = 011 = 3$ then we need to change p_3 so that its parity property is changed by the step 5.3.1) in hiding phase, we choose $p_u = p_3 = 0$ and then change it to $p'_3 = 0 + 1 = 1$. In the extracting phase, with the new $F = (12, 8, 1, 25, 84, 255, 45)$, $K = (0, 1, 1, 0, 0, 1, 0)$ and $V_F = (0, 0, 1, 1, 0, 1, 1)$, we get $T\text{new} = V_F \oplus K = (0, 1, 0, 1, 0, 0, 1)$ and therefore $m = 010 \oplus 100 \oplus 111 = 001 = d$; $x = (\lfloor 12/2 \rfloor + \lfloor 8/2 \rfloor + \lfloor 1/2 \rfloor + \lfloor 25/2 \rfloor + \lfloor 84/2 \rfloor + \lfloor 255/2 \rfloor + \lfloor 45/2 \rfloor)\ mod\ 2 = (6 + 4 + 0 + 12 + 42 + 127 + 22)\ mod\ 2 = 1 = d_4$. Then, $c = m + x \cdot 8 = 1001$ is the recovered value we need.

(c) $c = 0010$: then $d_4 = 0 \neq x$ and $d = 010 = m$, we choose one pixel p_k, such as $p_7 = 45$ and change p_7 to $p_7 - 2 = 43$, by the step 5.2.2) in hiding phase. In the extracting phase, with the new $F = (12, 8, 0, 25, 84, 255, 43)$, $V_F = (0, 0, 0, 1, 0, 1, 1)$, we get $T\text{new} = V_F \oplus K = (0, 1, 1, 1, 0, 0, 1)$ and therefore $m = 010 \oplus 011 \oplus 100 \oplus 111 = 010 = d$; $x = (\lfloor 12/2 \rfloor + \lfloor 8/2 \rfloor + \lfloor 1/2 \rfloor + \lfloor 25/2 \rfloor + \lfloor 84/2 \rfloor + \lfloor 255/2 \rfloor + \lfloor 43/2 \rfloor)\ mod\ 2 = (6 + 4 + 0 + 12 + 42 + 127 + 21)\ mod\ 2 = 0 = d_4$. Then, $c = m + x \cdot 8 = 0010$ is the recovered value we need.

(d) $c = 0100$: then $d_4 = 0 \neq x$ and $d = 100 \neq m$, set $u = d \oplus m = 110 = 6$, we need to change $p_6 = 255$ by $255 - 3 = 252$ so that its parity property is changed by the steps) in hiding phase. In the extracting phase, with the new $F = (12, 8, 0, 25, 84, 252, 45)$, $V_F = (0, 0, 0, 1, 0, 0, 1)$, we get $T\text{new} = V_F \oplus K = (0, 1, 1, 1, 0, 1, 1)$ and therefore $m = 010 \oplus 011 \oplus 100 \oplus 110 \oplus 111 = 100 = d$; $x = (\lfloor 12/2 \rfloor + \lfloor 8/2 \rfloor + \lfloor 1/2 \rfloor + \lfloor 25/2 \rfloor + \lfloor 84/2 \rfloor + \lfloor 253/2 \rfloor + \lfloor 45/2 \rfloor)\ mod\ 2 = (6+4+0+12+42+126+22)\ mod\ 2 = 0 = d_4$. Then, $c = m + x \cdot 8 = 0100$ is the recovered value we need.

4 Searching in hidden text with fuzzy automata approach

In this section, we present an application of the hiding scheme (2,14,8) for searching patterns in embedded text in stego-images of 256 gray levels. In some special cases or applications, one need to verify whether secret string or not, without revealing extracted hidden text in the images.

In Sect. 4.1, we recall one of our approach to application of finite automata technique in text searching area. In Sect. 4.2, we explain the way to apply automata technique to assign an algorithm for searching hidden texts in stego-images without revealing extracted texts in caches in insecurity environments.

4.1 Searching text with finite automata by fuzzy approach

Applications of finite automata and fuzzy automata have been given in the literature by many authors. Some of them can be seen in [15] for finite automata and [16–18] for fuzzy automata and fuzzy languages. In pattern matching problem, there are well-known algorithms: KMP and BM, where for given pattern text P and tagged text S, one reads the text S from left to right to verify and calculate the time that P is appeared in S. In KMP one, each letter S[i] of S is read in each step, $i = 1, \ldots, n$ and using this letter the automaton A_P assigned with P takes at least one state transition. The number of letters used in KMP algorithm is exact n, the length of S.

In BM algorithm, some time we can bypass some letters and therefore in real applications, BM algorithm runs faster than KMP (about 1, 7 in speed by our experiment results). For hidden text searching problem, we use the hiding scheme (2,14,8), for a stego-image G of 256 gray levels where this hidden text S is embedded. After getting step by step each letter S[i] from each 14-block of pixels from G, one letter in each time, S[i] is presented as a byte (8 bits), one immediately permute S[i] to $S[i]' = f(S[i])$ by a random permutation f and use this letter S[i] for a suited automaton so that after reading n letters $S[i]', i = 1, \ldots, n$, one can obtain the number of appearances of P in S, without saving any letter S[i] in caches where unknown attackers are waiting for.

To obtain such an automaton, say A_P, at first we explain the principle to the actions of this A_P by a fuzzy approach via a concrete example for P, S, at first in transitional case, and then show the structure of A'_P which is built from A_P by taking a secret permutation in detail.

For example, $P = abaca$, $S = gbaabacabacabha$. We assign a finite automaton A_P, each time A_P reads one letter $S[i], i = 1, 2, \ldots, 15$ and taking one state transition, at the same time it updates the fuzziness of appearance of the pattern text P in S. The fuzziness of appearance of P in S at the point S[i] can be considered as the longest prefix of P appeared from S[i] by backwards direction to S[1] and they are shown in the Table 1, they ranged from 0 to $5 = |P|$ (the length of P), here we need not normalize the fuzziness to [0; 1] since taking this calculation implies that the speed of the matching process has to slow down.

In Table 1, at each time $i, i = 1, \ldots, n$, if the automaton gains the fuzziness as 5, this means that at this point P appears in S one more time.

To obtain the automaton A_P, we show its structure in Fig. 1 and its state transitions in Table 2. For general case of P and S, the A_P can be obtained by automata technique in some similar way [15] to get automata for KMP algorithm, which we ignore for the restriction of our framework (Fig. 2).

Table 1 Fuzziness of P at each time i

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S[i]	g	b	a	a	b	a	c	a	b	a	c	a	b	h	a
Fuzziness of P	0	0	1	1	2	3	4	5	2	3	4	5	2	0	1

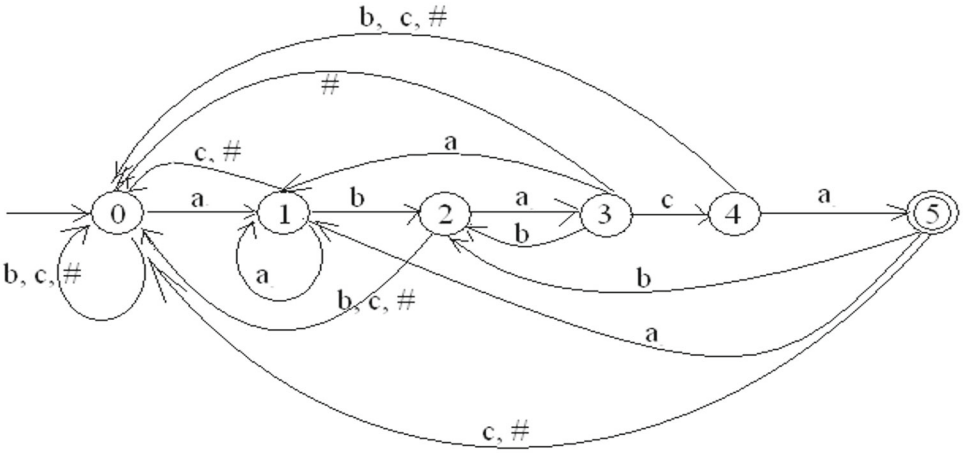


Fig. 1 Structure of A_P to calculate fuzziness of P in matching process where # stands for any letter not in P

Table 2 Transition law of the automaton A_p where # stands for any letter not in P

State	Read letter			
	a	b	c	c
0	1	0	0	0
1	1	2	0	0
2	3	0	0	0
3	2	0	4	0
4	5	2	0	0
5	1	2	0	0

4.2 Searching in hidden text embedded in stego-images

In this section, we use the above example to explain our idea, which is easy to generalize after and give a direct algorithm for matching in hidden texts. Suppose that after obtaining $P = abaca - a$ pattern text as the input, we generate randomly a permutation f to permute letters in the alphabet of texts. Suppose $f(a) = g$, $f(b) = d$, $f(c) = u$.. then we can reconstruct the A'_P from A_P by applying f to transition in A_P. The A'_P is given by Fig. 3.

Using this automaton A'_P, each time we extract a letter S[i] from the stego-image G, by immediately taking $S'[i] = f(S[i])$, one by one we get the encoded letters of the form $\#d\ g\ g\ d\ g\ u\ g\ d\ g\ u\ g\ d\#\ g$ which are processed via the A'_P with the fuzziness of P given by exact steps as in Table 3. Hence, the number of appearances of P in S is also calculated exactly as before.

Let us remark that each time a letter $S[i]$ is read, only the letter $S'[i]$ is saved in a memory variable which is always changed in each step, and each time reading a letter, the automaton A'_P takes only one state transition, not the same KMP one, therefore the speed of the whole matching process is faster than the KMP one, in case we do not care of extracting operations but only matching operations. In the comparative results between our algorithm with the KMP one, to obtain the runtime of matching phase for our method, we need to save $S'[1]\cdots S'[n]$ in a string buffer first and then calculate the runtime of the whole matching process. The experimental results show that the runtime is the same as the runtime of the BM algorithm, and they run faster than the KMP one with the ratio of about $1/7$ in speed.

5 Experimental results

We proposed the (1,7,4) scheme for data hiding. To prove our proposed scheme is correct, we performed an experiment to verify that it ensures the hidden image can be restored. In addition, our method is feasible for making good-quality stego-images from the original grayscale image. Thus, our proposed scheme has an ability to resist histogram detection algorithm from attackers. According to data hiding schemes, some schemes have many flipping LSB to conceal secret bits, the other have a few flipping to conceal the same bits.

Fridrich and her group presented a steganalytic method that does detect images with F5 content. They estimated the cover-image histogram from the stego-image and compared statistics from the estimated histogram against the actual histogram. As a result, they found it possible to get a modification rate β that indicates if F5 modified an image.

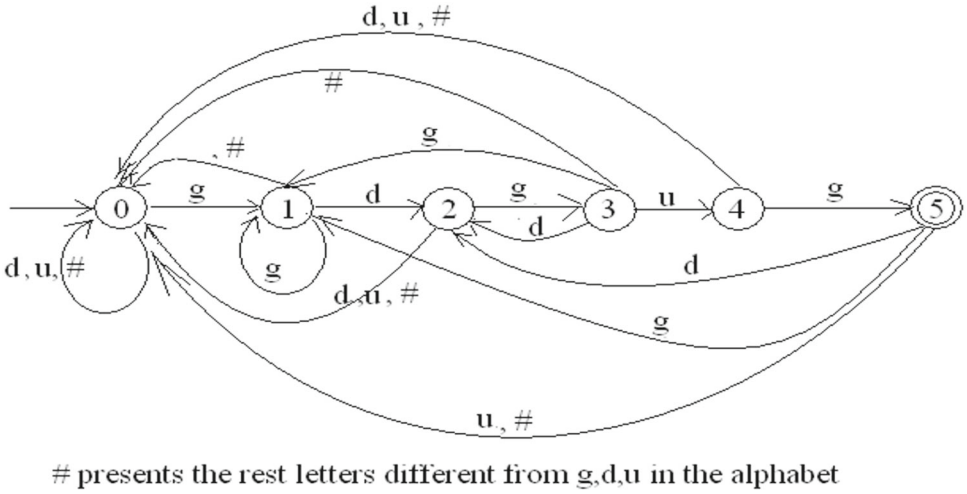
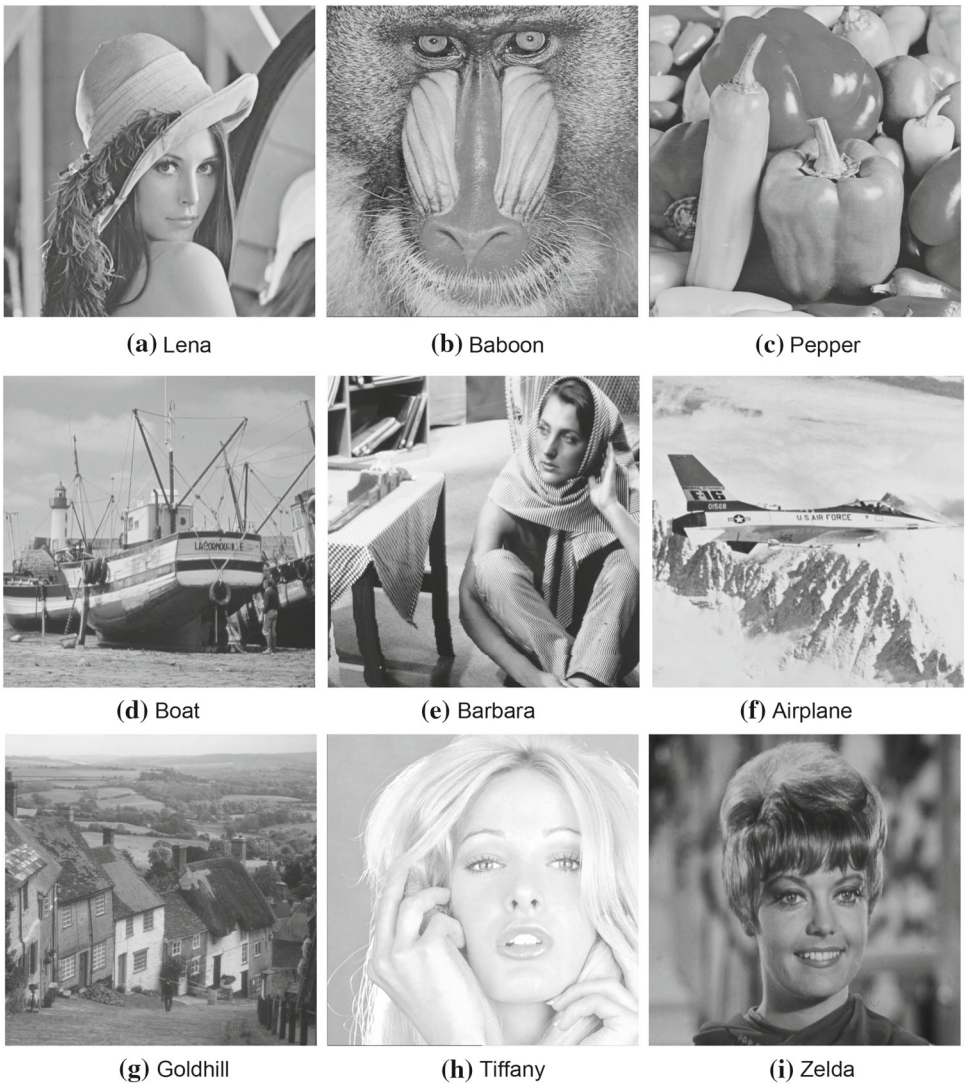


Fig. 2 Automaton A'_P after taking permutation on labels of A_P



(a) Lena **(b)** Baboon **(c)** Pepper

(d) Boat **(e)** Barbara **(f)** Airplane

(g) Goldhill **(h)** Tiffany **(i)** Zelda

Fig. 3 Original image for experiments

Table 3 Fuzziness of P at each time I using automaton A'_p

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S[i]	#	d	g	g	d	g	u	g	d	g	u	G	d	#	g
Fuzziness of P	0	0	1	1	2	3	4	5	2	3	4	5	2	0	1

Fig. 4 The setgo-image after hiding data by scheme (2,14,8)

To carry out our experiment, 512 × 512 grayscale images were used as cover images. Fig. 3 shows cover images for experiment to verify our proposed scheme. In our experiments, the qualities of the stego-images are measured by the peak-signal-to-noise ratio (PSNR) [9]. The PSNR is the most popular criterion for measuring distortion between the original image and shadow images. It is defined as follows:

$$\text{PSNR} = 10 \log_{10} \frac{255_2}{\text{MSE}} \tag{1}$$

where MSE is the mean square error between the original grayscale image and the shadow image:

$$\text{MSE} = \left(\frac{1}{N}\right)^2 \sum_{k=1}^{M} \sum_{k=1}^{N} (x_{ij} - x'_{ij})^2 \tag{2}$$

The symbols x_{ij} and x'_{ij} represent the pixel values of the original grayscale image and the stego-image at position (i, j), respectively; M and N are the width and height of the original image, respectively.

$$p = \frac{|\delta|}{m \times n}(bpp) \tag{3}$$

In Eq. (3), p denotes bits-per-pixel (bpp) where m and n are the width and height of the original image, respectively, and Eq. (3) is an embedding payload.

Our experiment compares how many secret bits can be carried by a cover pixel. δ is the number of bits of a secret message δ. There is a tradeoff between a payload and quality of an image. To increase the embedding rate, it is too obvious to require a sacrifice of image quality. We apply the hiding scheme (2,14,8) for 256 gray level image "Lena" 512 × 512 pixels. In Figs. 3 and 4, the original and setgo-images "Lena" are shown. After experiment of (2,14,8) scheme, PSNR is 56.066 dB and embedded bytes are 18,720 bytes. As a result, the hidden ratio is about 1/14, so our proposed (2,14,8) scheme proved that it is very strong to resist detection from stego images.

Table 4 Payloads for scheme and improving scheme (1,7,4)

Images	Matrix coding [21]		Hamming+1 [10]		(1,7,4) scheme (Proposed)	
	PSNR	p	PSNR	p	PSNR	p
Baboon	56.44	0.43	53.71	0.499	50.87	0.57
Barbara	54.65	0.43	48.60	0.499	50.48	0.57
Boats	54.75	0.43	49.37	0.499	50.51	0.57
Goldhill	57.02	0.43	53.73	0.499	50.49	0.57
Airplane	55.84	0.43	51.61	0.499	50.48	0.57
Lena	56.05	0.43	52.43	0.499	50.91	0.57
Pepper	54.01	0.43	47.26	0.499	50.50	0.57
Tiffany	53.98	0.43	47.46	0.499	50.53	0.57
Zelda	56.40	0.43	54.04	0.499	50.47	0.57
Average	56.44	0.43	50.91	0.499	50.58	0.57

Table 4 shows the visual quality of the stego-images created by the matrix encoding, "Hamming+1", and "(1,7,4) scheme". Our proposed "(1,7,4) scheme" shows 0.57 bpp with a good visual quality (i.e., the PSNR value is higher than 50.58 dB). From Table 4, for the visual quality factor, the "(1,7,4) scheme" shows a higher visual quality outcome. For embedding payload comparison, the proposed "(1,7,4) scheme" as a high embedding payload outcome. In this experiment, we verified that "(1,7,4) scheme" is worth the steganography method, because our scheme shows reasonable embedding rate and quality as a data hiding scheme. As the PSNR of our scheme is over 50.58 dB, it is not easily detectable by attackers. Therefore, our scheme is highly suitable for various fields of steganography.

6 Conclusion

In this paper, in the first problem of hiding secret data in image, from a hiding scheme (1,7,3) built on Z_2-module, we extend to a new good hiding scheme (1,7,4) for 8-bit grayscale images which permits us to reach the theoretical limit $MSDR_1$ and one can use it in real applications. Directly, a hiding scheme (2,14,8) for 8-bit grayscale images is obtained. This scheme is suited to solve the second problem for searching pattern text in hidden text which is embedded in an 8-bit grayscale image. The matching method is based on automata technique with fuzzy approach which permits us to establish a matching text algorithm in practical having the runtime approximated to the runtime of well-known BM-matching text algorithm.

An interesting question is remained open: whether there exists a hiding scheme (2,14,9) for 8-bit grayscale images in which the differences of changed gray values to the original ones do not exceed 3.

Acknowledgments This research was supported by Basic Science Research Program Through the National Research Foundation of Korea (NRF) by the Ministry of Education, Science and Technology (20120192). We are grateful to Dr. Phan Trung Huy for supporting this research. He was a good researcher and best professor at HUST.

References

1. Anderson, R.J., Petitcolas, F.A.P.: On the limits of steganography. IEEE J. Sel. Areas Commun. **16**(4), 474–481 (1998)
2. Kim, C., Yang, C.-N.: Watermark with DSA signature using predictive coding. Multimed. Tools Appl. 1–15 (2013). doi:10.1007/s11042-013-1667-6
3. Machado, R.: EzStego. http://www.stego.com
4. Fridrich, J.: A new steganographic method for palette-based images. In: Proceedings of the Conference on Image Processing. Image Quality and Image Capture Systems (PICS-99), Savannah, Georgia, USA, 25–28 April 1999, 285–289
5. Westfield, A., Pfitzmann, A.: Attacks on steganographic systems breaking the steganography utilities EzStego, Jsteg, Steganos and S-Tools and some lessons learned. Third International Workshop, IH'99 Dresden Germany, Computer. Science. **1768**, 61–76 (1999)
6. Kawaguchi, E., Eason, R.: Principle and applications of BPCS-Steganography. In: Proceedings of SPIE 3528, Multimedia Systems and Applications, 464. 3528, (1998)
7. Mielikainen, J.: LSB matching revisited. IEEE Signal Process. Lett. **13**(5), 285–287 (2006)
8. Zhang, X., Wang, S.: Efficient steganographic embedding by exploiting modification direction. IEEE Commun. Lett. **10**(11), 781–783 (2006)
9. Kim, C.: Data hiding by an improved exploiting modification direction. Multimed. Tools Appl. **69**(3), 569–584 (2014)
10. Chang, C.C., Kieu, T.D., Chou, Y.C.: A high payload steganographic scheme based on (7, 4) Hamming code for digital images. International Symposium on Electronic Commerce and Security, 16–21 (2008)
11. Zhang, W., Wang, S., Zhang, X.: Improving hiding efficiency of covering codes for applications in steganography. IEEE Commun. Lett. **11**(8), 680–682 (2007)
12. Huy, P.T., Thanh, N.H.: On the maximality of secret data ratio in CPTE schemes. Lect. Notes Comput. Sci. **6591**, 88–99 (2011)
13. Huy, P.T., Thanh, N.H., Kim, C.: Relationship between correcting code and module technique in hiding secret data. Netw. Parallel Comput. Lect. Notes Comput. Sci. **7513**, 297–307 (2012)
14. Huy, P.T., Kim, C., Anh, L.T., Hoa, L.Q., Yang, C.N.: Data hiding based on palette images using weak bases of Z_2-modules. In: Grid and Pervasive Computing (GPC). Lecture Notes in Computer Science, **7861**, 649–658 (2013)
15. Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**(2), 323–350 (1977)
16. Lee, E.T.: Application of fuzzy languages to pattern recognition. Kybernetes. **6**(3), 167–173 (1977)
17. Mordeson, J.N., Malik, D.S.: Fuzzy Automata and Languages: Theory and Application. Chapman and Hall/CRC (2002) (ISBN:978-1584882251)
18. Peter, R.J.: A bibliography on fuzzy automata, grammars and languages. Int. J. Comput. Math. INF-95-46, 1–10 (1995)
19. Tseng, Y.-C., Chen, Y.-Y., Pan, H.-K.: A secure data hiding scheme for binary images. IEEE Trans. Commun. **50**(80), 1227–1231 (2002)
20. Westfeld, A.: High capacity despite better steganalysis (F5-A Steganographic Algorithm). Information Hiding, 4th International Workshop, Lecture Notes Computer. Science. **2137**, 289–302 (2001)
21. Westfeld, A.: F5: a steganographic algorithm. In: Proceedings of 4th International Workshop Information Hiding 2001, Lecture Notes in Computer Science, 2137(1), 289–302 (2001)